Campus de l'Université de Montréal
C.P. 6128, succursale Centre ville
Montréal QC H3C 3J7 Canada

Telephone (514) 343 7575
Telecopieur (514) 343 7121

crt@crt.umontreal.ca
www.crt.umontreal.ca

CRT-2001-44

Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

GENERATING TRADING RULES ON THE STOCK MARKETS WITH GENETIC PROGRAMMING

by

Jean-Yves Potvin
Patrick Soriano
Maxime Vallée

November 2001

103-03076/

PUBLICATION
CRT-2001-44

GENERATING TRADING RULES ON THE STOCK MARKETS WITH GENETIC PROGRAMMING*

by

Jean-Yves Potvin[1,2]
Patrick Soriano[1,3]
Maxime Vallée[4]

* This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and by the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR).

1 Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7
2 Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7
3 Service de l'enseignement des méthodes quantitatives de gestion, École des Hautes Études Commerciales, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7
4 Service de l'enseignement de la finance, École des Hautes Études Commerciales, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2001

Abstract. Technical analysis is aimed at devising trading rules capable of exploiting short-term fluctuations on the financial markets. Recent results indicate that this market timing approach may be a viable alternative to the buy-and-hold approach, where the assets are kept over a relatively long time period. In this paper, we propose genetic programming as a means to automatically generate such short-term trading rules on the stock markets. Rather than using a composite stock index for this purpose, the trading rules are adjusted to individual stocks. Computational results, based on historical pricing and transaction volume data, are reported for fourteen Canadian companies listed on the Toronto stock exchange market.

Key words. Stock markets, technical analysis, genetic programming, trading rules.

Résumé. L'analyse technique vise à développer des règles d'achat et de vente de titres financiers permettant de prendre avantage des fluctuations à court terme dans les marchés. Des résultats récents indiquent qu'il s'agit d'une alternative viable à la stratégie classique, connue sous le nom de "buy-and-hold", où les titres sont conservés sur une période relativement longue. Dans cet article, nous proposons la programmation génétique afin de générer automatiquement de telles règles d'achat et de vente, en nous concentrant sur des titres individuels plutôt que sur des indices composites. Des résultats expérimentaux, obtenus à partir de données historiques portant sur des prix et des volumes de transactions, sont présentés pour quatorze compagnies canadiennes inscrites à la bourse de Toronto.

Mots-clés. Marchés financiers, analyse technique, programmation génétique, règles d'achat et de vente.

1. Introduction

In stock exchange markets, the "buy-and-hold" approach is a well-known strategy among traders. Basically, if a company and its activity sector look promising, the trader buys and keeps his assets over a relatively long time period. An alternative approach, known as market timing, is more dynamic and focuses on short term fluctuations. Through technical analysis, trading rules are devised to generate appropriate buying and selling signals over short time periods. The purpose of this paper is to demonstrate that genetic programming, a recent development in the field of evolutionary algorithms, can be exploited to automatically generate such trading rules.

There is a fairly large literature related to technical analysis in various financial domains. Although the first results in the 60's and 70's did not demonstrate any benefits over the traditional buy-and-hold approach (Alexander 1964, Fama and Blume 1966, Fama 1970, Jensen and Bennington 1970), some recent developments are more promising. For example, Pruitt and White (1988) developed the CRISMA trading system which showed positive returns over a 10-year period, based on transaction costs of 2%. Brock et al. (1992), followed by Bessembinder and Chan (1995), also demonstrated that simple trading rules could be profitable (but, without transaction costs). Other successful applications of technical analysis in the currency exchange market may be found in Sweeney (1986), Levich and Thomas (1993) and Osler and Chang (1995).

Although interesting, these developments are based on a priori rules determined through technical analysis. The emergence of new technologies, in particular evolutionary algorithms, now allows a system to automatically generate and adapt trading rules to a particular application. Genetic algorithms (Holland 1975), for example, have already been applied to a number of financial applications, like those mentioned in Bauer (1994). For learning trading rules, however, the genetic programming (GP) approach of Koza (1992) looks more promising, as it provides a flexible framework for adjusting the trading rules (which may be seen as 'programs') to the current environment. Although, the first attempts by Chen and Yeh (1997) and Allen and Karjalainen (1999) on the stock exchange markets did not show any excess returns with regard to the buy-and-hold approach, other applications of GP on the currency exchange markets, where the transaction costs are very low, have proven otherwise (Neely et al. 1997, Neely and Weller 1999).

Our goal here is to explore once more the application of GP on stock exchange markets, but to consider stocks offered by individual companies, rather than global market indices (e.g., Dow Jones, S&P 500), as it is done in previous studies. This approach looks more promising, given that each rule generated through GP will now be adjusted to an individual stock or activity sector. Also, the sale of stocks which the seller does not own will be allowed to take advantage of falling stock prices (i.e., by later purchasing the stocks at a lower price, a profit is made). This has never been done in the past because the implementation of simultaneous short sales in the case of a composite stock index would have been very difficult to realize. Another goal of this study is to identify markets where the trading rules generated with GP are particularly indicated.

The remainder of the paper is the following. In the next section, we first introduce the GP paradigm. Then, its adaptation to our application domain is presented in Section 3. Computational results obtained with the stocks of fourteen Canadian companies operating in different activity sectors are reported in Section 4. The conclusion follows.

2. Genetic Programming

Genetic programming (Koza 1992, Koza 1994, Koza et al. 1999) is a recent development in the field of evolutionary algorithms which extends classical genetic algorithms by allowing the processing of non linear structures. A genetic algorithm (Holland 1975, Goldberg 1989, Davis 1991, Chambers 1995, Michalewicz 1996) is a randomized search procedure working on a population of individuals or solutions encoded as linear bit strings. This population evolves over time through the application of operators which mimic those found in nature, namely, selection of the fittest, crossover and mutation. First, "parent" chromosomes are selected in the population for reproduction. The selection process is probabilistic and biased towards the best individuals (to propagate good solution features to the next generation). Then, the crossover operator combines the characteristics of a pair of parent chromosomes to create two new offspring, in the hope that these offspring will be better fit than their parents. An example of one-point crossover is shown in Figure 1, where the cross point is randomly chosen between the second and third bit. In this case, the end parts of the two parent chromosomes are exchanged to create the offspring.

1	0	|	1	0	0	(parent 1)
0	**0**	|	**1**	**1**	**1**	(parent 2)
1	0		**1**	**1**	**1**	(offspring 1)
0	**0**		1	0	0	(offspring 2)

Figure 1. One-point crossover on two bit strings

Finally, the mutation operator is applied to each offspring. This operator processes the offspring position by position and flips the bit value from 0 to 1, or from 1 to 0, with a small probability at each position. Mutation is viewed as a "background" operator which slightly perturbs a small proportion of solutions. Its primary goal is to maintain or restore diversity in the population and to guarantee that every state of the search space is accessible from the current state. Through the selection/crossover/mutation process, it is expected that an initial population of randomly generated individuals will improve as parents are replaced by better offspring.

Genetic programming extends the above paradigm by allowing the evolution of programs encoded as tree structures. These programs are constructed from a predefined set of functions and terminals (which may be variables, like the state variables of a particular system, or constants, like integer *3* or boolean *False*). The evolution of programs within the genetic programming framework can be summarized as follows:

Step 1. *Initialization.* Create an initial random population of P programs and evaluate the fitness of each program by applying it on a set of *fitness cases* (examples). Set the current population to this initial population.

Step 2. *Selection.* Select P programs in the current population (with replacement). The selection process is probabilistically biased in favor of the best programs.

Step 3. *Modification.* Apply reproduction or crossover to the selected programs.

Step 4. *Evaluation.* Evaluate the fitness of each offspring in the new population.

Step 5. Set the current population to the new population of offspring.

Step 6. Repeat steps 2 to 6 for a predefined number of generations or until the system does not improve anymore.

The final result is the best program generated during the search. The main issues related with this algorithm are briefly discussed in the next subsections.

2.1 Encoding

The individual structures which evolve over time are computer programs represented as tree structures. In our context, these structures will encode trading rules. The size, shape and contents of the trees, which are recursively constructed from a predefined set of functions G and terminals T, dynamically change during the evolution process. Each particular function g_i in the function set takes a specified number of arguments and should be able to accept as argument the value returned by the other functions in the set. Figure 2 illustrates a program with $G = \{+, *\}$ and $T = \{C_0, C_1\}$. Such a program computes $C_0 \times C_1 + C_0$, where C_0 and C_1 are numerical constants.

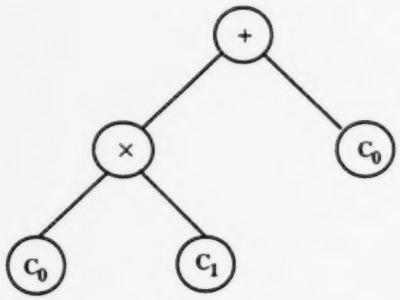

Figure 2. A tree structure for a program

2.2 Fitness evaluation

The fitness of a program is evaluated by applying the program to a set of fitness cases. In our context, the excess return provided by a trading rule over the buy-and-hold approach is evaluated using historical data.

2.3 Initial population

Typically, the initial population is made of randomly generated programs or trees. Usually, an equal mix of trees is produced with the *Full* and *Grow* methods, which are defined as follows.

Full. This method creates trees such that the length of every path between a terminal and the root is *equal* to a predefined depth.

Grow. This method creates trees of variable shapes. That is, the length of a path between a terminal and the root is *at most* a predefined maximum depth.

2.4 Selection

Different methods have been reported in the literature for selecting the programs that either reproduce or are transformed by crossover. The basic idea is to bias the selection process towards the best programs through a so-called fitness-proportionate selection scheme, where the selection probability prb_i of program i is proportional to its fitness f_i. Namely:

$$prb_i = \frac{f_i}{\sum_{j=1}^{P} f_j}.$$

In this equation, the denominator sums the fitness values over all programs in the current population.

2.5 Reproduction and crossover

A new generation of programs is obtained through the action of reproduction and crossover. Other secondary operators can also be applied to the programs, like mutation.

2.5.1 Reproduction

Reproduction simply makes a copy of the selected program in the new population, without any modification

2.5.2 Crossover

Crossover applies to two selected programs. A crossover point, namely a function or terminal within the tree, is randomly selected within each parent. This crossover point defines a fragment which is the subtree consisting of the crossover point plus the entire subtree lying below the crossover point. Two new offspring programs are then produced by exchanging the two fragments, as depicted in Figure 3. In this example, the mathematical operators / and × are selected in the first and second tree, respectively. Then, the subtrees are exchanged to create two new programs. Obviously, when both crossover points are terminals (i.e., leaves of their respective tree), the effect of crossover is simply to exchange the two terminals. Typically, the probability distribution for the selection of a crossover point favors the exchange of subtrees rather than terminals.

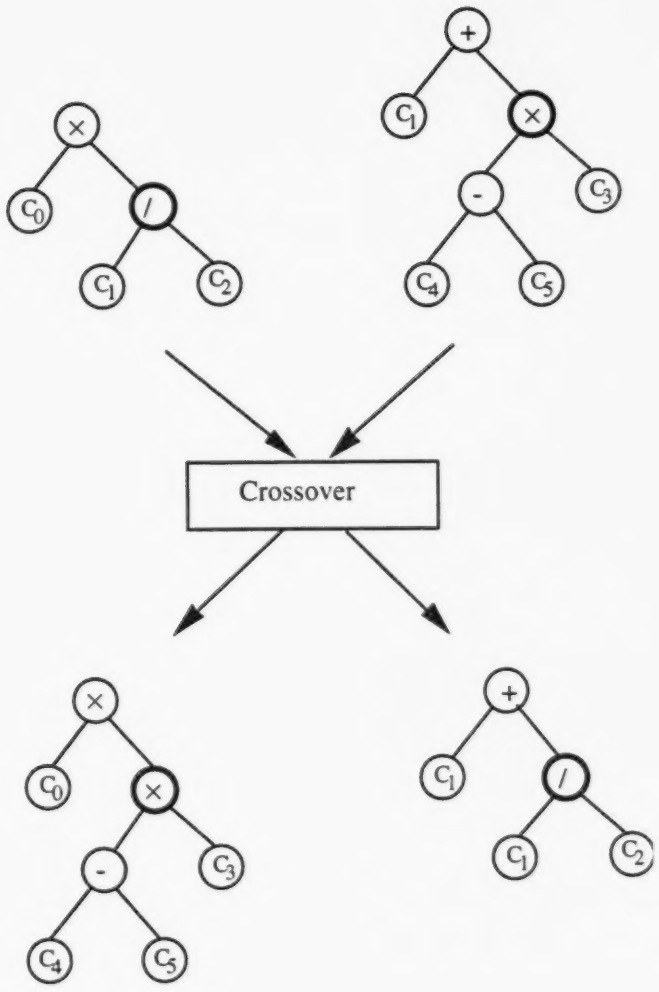

Figure 3. Crossover in genetic programming

2.6 Mutation

Mutation is a unary operator aimed at restoring diversity in a population by applying random modifications to individual structures. Given that the trees manipulated by GP are more complex than the linear structures of a GA, the loss of diversity is not a primary concern here. In particular, the crossover of two identical programs is unlikely to lead to offspring which are identical to their parents, as long as the cross points are different on both parents. The mutation operator is defined as follows in a GP. First, a

random point is chosen in the tree. This point, as well as the subtree below it, are then removed and replaced by a new, randomly generated, subtree.

3. Generating trading rules with GP

Based on the description provided in the previous section, the adaptation of GP in our context is now presented. The GP will generate trading rules encoded as programs or tree structures. These rules are boolean functions which return either a buy (True) or a sell signal (False). If, at the end of the current day, the position on the market is 'open' and a buy signal is sent by the rule, the stock is bought at the opening of the market the next day and the position switches to 'close'. Conversely, if the current position is 'close' and a sell signal is sent, the stock is sold and the position switches to 'open'. Otherwise, the current position does not change.

3.1 Encoding

The set of functions G is made of classical arithmetic, boolean and relational operators, plus the boolean function if-then-else and a number of real functions. The set of terminals T correspond to numerical and boolean constants, plus the variables *price* and *volume*. They are described in the following.

Functions

Arithmetic operators : +, -, /, ×;
Boolean operators : and, or, not;
Relational operators : <, >.

Boolean functions : if-then-else.

Real functions
- norm(r_1,r_2) : absolute value of the difference between two real numbers;
- avg(s,n) : average of price or volume over the past n days;
- max(s,n) : maximum value of price or volume over the past n days;
- min(s,n) : minimum value of price or volume over the past n days;
- lag (s,n) : price or volume is lagged by n days;
- volatility(n) : variance in daily returns over the past n days;
- RSI(n) : relative strength index;
- ROC(n) : rate of change.

The last two functions, which are often used in technical analysis, are defined as:

$$ROC(n) = \left(\frac{\text{closing price of current day}}{\text{closing price } n \text{ days ago}} - 1 \right) \times 100, \text{ and}$$

$$RSI(n) = 100 - \left(\frac{100}{1 + RS(n)} \right), \text{ where } RS(n) = \frac{\sum_{i \in D^+(n)} r_i}{-\sum_{i \in D^-(n)} r_i}$$

and where $D^+(n)$ is the set of days (over the past n days) with rising stock prices, $D^-(n)$ is the set of days (over the past n days) with falling stock prices and r_i is the return of day i, which is positive when the stock price is rising and negative otherwise.

Terminals

Constants

Real	: chosen in the interval [0,250], where 250 is the approximate number of working days in a year;
Boolean	: True, False;
User-defined	: price, volume;

Real variables

p	: stock price of the current day;
v	: transaction volume of the current day;

Clearly, the functions and terminals chosen to construct the trading rules violate the closure assumption of GP. That is, a given function cannot necessarily accept as argument the value returned by another function in set G, given that both boolean and real functions are found in that set. This particularity imposes restrictions on the structure of the trading rules. Consequently, real functions are always found in the lower part of the tree, boolean functions and operators in the upper part (including the root) and relational operators make the transition between the two. A relational operator may also be found at the root of the tree. A typical tree representing a trading rule is illustrated in Figure 4. In this example, the rule sends a buy signal if the average stock price over the past 50 days is greater than the current price or the current transaction volume is less than 20.

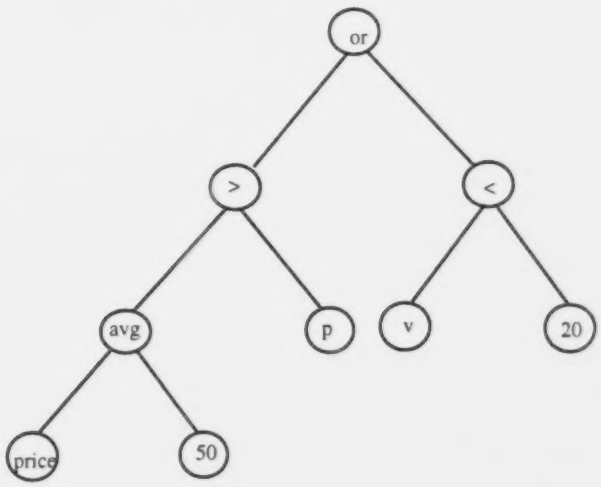

Figure 4. Example of a trading rule

3.2 Fitness evaluation

The raw fitness of a trading rule is its excess return over the buy-and-hold approach, as evaluated on historical data. For an investment period extending from day 0 to day t, the return of the buy-and-hold approach is simply the price of day t minus the price of day 0. In the case of the trading rules generated by GP, the gains or losses (if any) are evaluated on a daily basis and summed up. The latter minus the former correspond to the excess return of the trading rule.

3.3 Initial Population

A mix of trees produced by the Full and Grow methods is used. Due to the particular structure of our trees, the recursive construction process (from the root to the leaves) must observe the following guidelines.

(1) The root of the tree is selected among the boolean functions and operators.

(2) Once the root has been selected, its descendants may be selected among boolean constants, boolean functions, boolean or relational operators.

(3) When a relational operator is selected, its descendants must be selected among real functions or terminals.

In the case of the *Full* method, the trees must be fully developed to a predefined depth, hence terminals can only be selected when that depth is reached. In the case of the *Grow* method, the trees do not need to be fully developed and terminals can be selected at any level (although terminals must be chosen when the predefined depth is reached).

3.4 Selection

In this work, a rank-based method is used (Baker 1985, Whitley 1989). First, all programs in the current population are sorted from best (rank 1) to worst (rank P) based on their "raw" fitness value. A new fitness value f_i is then associated with the program of rank i as follows:

$$f_i = Max - \left[(Max - Min) \frac{i-1}{P-1} \right]$$

Hence, the best program gets fitness *Max*, the worst program gets fitness *Min* and the fitness values of the remaining programs are equally spaced between *Min* and *Max*. In our experiments, *Max* and *Min* were set to 1.8 and 0.2, respectively. For example, when the population is composed of P=5 programs, these programs are evaluated from best to worst as follows: f_1=1.8, f_2=1.4, f_3=1.0, f_4=0.6 and f_5=0.2. Stochastic universal sampling (Baker 1987), a variant of fitness-proportionate selection where all parents are selected at once in a single trial, is then invoked on the transformed fitness values.

The ranking method basically modifies the raw fitness values before stochastic universal sampling is called upon. Otherwise, a "super-program" with high fitness would

typically be overselected (leading to premature convergence on a possibly suboptimal solution). Conversely, a random selection process would occur when all programs have similar fitness values. Through ranking, the gap between very close fitness values is enlarged, thus alleviating the random selection behavior, while the impact of a super-program is greatly reduced by bounding its fitness to the *Max* value.

3.5 Reproduction and crossover

The classical reproduction and crossover operators presented in Section 2 were used in this study. Due to the particular layered structure of our trees (see subsection 3.1), a special care was given to the crossover operator. Once a cross point is chosen on the first parent, the subset of compatible cross points is identified on the other parent and only this subset is considered for crossover. If there is no compatible cross point, two other parents are selected.

4. Computational results

In this section, we present results obtained with our GP on stocks of Canadian companies. First, the data at our disposal are described in subsection 4.1. Then, subsection 4.2 presents the parameter settings for the GP. Section 4.3 reports numerical results obtained with each stock.

4.1 Data

For the study, Canadian companies were chosen from the TSE 300 index which spans fourteen different activity sectors. One company was selected in each sector, but only among those which were active on the market before 1990. The companies used for the study are listed in Table 1. The historical data included the stock price and the transaction volume for each working day between June 30, 1992 and June 30, 2000, for a total of 2003 days.

Activity sector	Company	Symbol
Mining	Alcan Aluminium Ltd	*AL*
Precious metals	Barrick Gold Corporation	*ABX*
Oil and gas	Cdn. Occidental Petroleum Ltd	*CXY*
Forestry	Abitibi-Consolidated Inc	*A*
Consumer products	Molson Inc	*MOL*
Industrial products	Bombardier Inc	*BBD*
Real estate	Cambridge Shopping Centers	*CBG*
Transportation and Env.	Trimac Corporation	*TMA*
Pipelines	TransCanada Pipelines Ltd	*TRP*
Utilities	Canadian Utilities Ltd	*CU*
Communications	The Seagram Company Ltd	*VO*
Merchandising	Sears Canada Inc	*SCC*
Financial services	Royal Bank of Canada	*RY*
Diversified (conglomerate)	Canadian Pacific Ltd	*CP*

Table 1. Canadian companies in the study

For the experiments, we considered both a long training period and a short one. In each case, an initial period of 250 days was put aside to start the process (given that some of the primitive functions used by the trading rules, such as *avg*, may require data from the last 250 days in the worst case, see subsection 3.1). The trading rules obtained on the training period were then evaluated on previously unseen data associated with a testing period. The exact time periods used for the experiments are reported below.

Short training period.

Initial (for training)	:	June 25, 1997 to June 22, 1998 (250 days)
Training	:	June 22, 1998 to June 25, 1999 (256 days)
Initial (for testing)	:	July 07, 1998 to June 25, 1999 (250 days)
Testing	:	June 28, 1999 to June 30, 2000 (256 days)

Long training period.

Initial (for training)	:	June 30, 1992 to July 02, 1993 (250 days)
Training	:	July 02, 1993 to June 25, 1999 (1498 days)
Initial (for testing)	:	July 07, 1998 to June 25, 1999 (250 days)
Testing	:	June 28, 1999 to June 30, 2000 (256 days)

4.2 Parameter settings

Preliminary experiments were performed to determine the best parameter settings for the GP. Based on these experiments, the parameter values shown in Table 2 were finally selected. Tests with populations of size 100, 250 and 500 were performed. In all cases, the best results were achieved with the largest populations. However, the computation times also increased from an average of five minutes for P=100 to more than one hour for P=500. Thus, populations with more than 500 individuals were not considered. The number of generations was fixed to 50, as no significant improvement was observed beyond that point. In fact, most of the improvement was achieved before generation 40 (see Table 5 in subsection 4.3). Apart from the maximum number of generations, the GP was also stopped after 15 consecutive generations without any improvement to the best solution. The other parameter values correspond to standard values suggested in Koza (1992).

Population size	500
Number of generations	50
Initialization method	Equal mix of Full and Grow
Selection method	Ranking + SUS
Reproduction rate	0.35
Crossover rate	0.60
Mutation rate	0.05
Probability of choosing an internal point (crossover)	0.90
Probability of choosing a terminal (crossover)	0.10
Maximum depth of programs in initial population	6
Maximum depth of programs in following generations	17

Table 2. Parameter settings

4.3 Numerical Results

Computational results were run on a 300 MHz Pentium II PC. The results obtained with GP on the 14 stocks for the short and long training periods are shown in Tables 3 and 4. For each stock, the numbers shown are the average of 10 different runs. As expected, the results obtained on the training period are much better than those obtained on the testing period. In fact, the overall average return on the testing period is negative for both the short and long periods, which indicates that the trading rules do not provide any improvement over the buy-and-hold approach. The positive returns on the training data for the short period (131.17%) are much higher than those obtained for the long period (22.93%), although the results are very similar when the trading rules are evaluated on the testing period. Due to the smaller amount of data in the case of the short training period, the GP might have developed trading rules more tightly fitted to those data. An intermediary control period was thus introduced to stop the training when a new best rule on the training period degraded the excess returns by more than 25% on the control period or when a degradation was observed for three consecutive iterations. Although this additional control mechanism allowed the GP to stop earlier in many cases, no significant improvement was observed on the testing period.

Symbol	Training	Testing
A	144.16%	15.29%
ABX	202.16%	38.12%
AL	135.95%	15.79%
BBD	123.04%	-34.92%
CBG	174.01%	-75.09%
CP	120.70%	0.86%
CU	56.69%	7.88%
CXY	183.40%	-102.16%
MOL	99.23%	17.38%
RY	125.68%	3.25%
SCC	110.80%	-21.84%
TMA	211.93%	42.27%
TRP	85.78%	36.04%
VO	146.78%	-10.83%
Average	131.17%	-4.85%
Number of positive returns	14	9

Table 3. Numerical results for the short training period

Symbol	Training	Testing
A	31.00%	18.55%
ABX	34.27%	7.69%
AL	21.99%	19.83%
BBD	18.30%	-79.46%
CBG	33.15%	-35.58%
CP	19.67%	12.11%
CU	9.77%	6.45%
CXY	30.39%	-44.85%
MOL	19.36%	13.74%
RY	19.59%	8.13%
SCC	16.53%	-7.33%
TMA	30.57%	54.80%
TRP	16.57%	21.09%
VO	19.88%	-45.41%
Average	22.93%	-3.59%
Number of positive returns	14	9

Table 4. Numerical results for the long training period

The optimization capabilities of the GP cannot be questioned, however, as illustrated in Table 5. In this table, we show the best and average trading rules in the initial population, as well as the best and average trading rules in the final population. We also indicate the generation number where the best trading rule was found. These results correspond to those observed with the long training period. From the start to the end of the GP, the best rule has improved on average from 11.22% to 22.93%, while the average rule in the population has improved from –8.64% to 16.19%.

The disappointing results of Tables 3 and 4 hide some interesting findings. First, 9 stocks out of 14 show positive returns on the testing period. There are five problematic stocks, BBD, CBG, CXY, SCC and VO, with some of them having a significant impact on the overall results. For example, without CBG and CXY which stand at -75.09% and –102.16% in Table 3, a positive average return of 9.1% is observed on the twelve other stocks. The poor results obtained on these two stocks can be explained by important dissimilarities in the transaction volumes and prices over the training and testing periods (i.e. relatively stable market during the training period versus a rising market during the testing period).

By collecting the results obtained over all runs on the testing period, the graph of Figure 5 was obtained. It shows the excess returns generated by the various trading rules over the buy-and-hold, as compared with the returns of the buy-and-hold alone. Clearly, it is difficult for the trading rules to do better than the buy-and-hold when the market is

rising, and conversely. For example, the Canadian Occidental Petroleum (CXY) stock exploded by 80.45% during the testing period. In these circumstances, the buy-and-hold approach is highly profitable and can hardly be outperformed.

Symbol	Generation	Initial Population		Final Population	
		Best	Average	Best	Average
A	37	15.08%	-3.53%	27.48%	21.89%
ABX	42	20.76%	3.28%	31.76%	27.13%
AL	36	8.53%	-10.32%	20.03%	14.93%
BBD	36	3.50%	-37.89%	14.80%	8.53%
CBG	37	24.73%	10.86%	31.90%	28.38%
CP	34	7.34%	-8.80%	17.36%	12.13%
CU	37	2.57%	-10.21%	8.72%	5.16%
CXY	37	18.02%	-7.09%	29.23%	24.28%
MOL	38	9.73%	-2.44%	17.51%	13.61%
RY	33	8.70%	-14.91%	17.85%	12.44%
SCC	40	4.90%	-24.99%	14.50%	9.66%
TMA	37	18.69%	2.26%	29.17%	24.09%
TRP	39	7.41%	-3.73%	14.62%	11.45%
VO	40	7.09%	-13.42%	17.78%	12.94%
Average	37	11.22%	-8.64%	22.93%	16.19%

Table 5. Evolution of the trading rules over the long training period

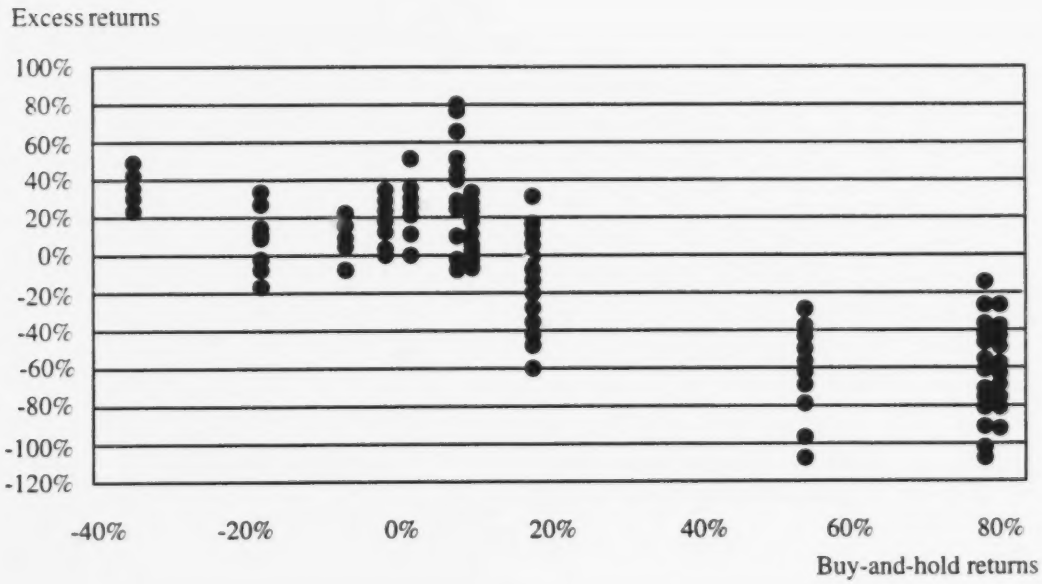

Figure 5. Excess returns vs buy-and-hold returns on the testing periods

The most interesting feature in the graph of Figure 5 is the performance of the trading rules in situations where the buy-and-hold approach generates returns which are close to 0%. In these cases, the excess returns generated by the trading rules are generally positive. It means that in a relatively stable market, the trading rules generated by GP are profitable.

4.4 Trading rules

As noted in previous studies (Benyahia and Potvin 1998), the trading rules produced through GP are rather intricate and do not provide useful insights about the decision problem. Even after eliminating redundancies, the decision trees contain more than 50 nodes on average, distributed over 5 or 6 hierarchical levels. Consequently, these decision trees can hardly be interpreted in a natural way. This is a weakness of this approach, although it is shared by other adaptive algorithms as well (e.g., connection weights in neural network models).

5. Conclusion

This paper has described an application of the GP paradigm for generating trading rules on the stock markets. The results are mixed, but show that the trading rules generated by the GP are beneficial when the market is relatively stable. On the other hand, the GP can hardly do better than the buy-and-hold approach when the market is rising. Hence, the idea of automatically generating trading rules for particular stocks, although highly attractive, should be used with care and only under specific market conditions.

Acknowledgments. Financial support was provided by the Canadian Natural Sciences and Engineering Research Council (NSERC) and by the Quebec Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR). This support is gratefully acknowledged.

References

Alexander S.S., "Price Movements in Speculative Markets : Trends or Random Walks", in The Random Character of Stock Market Prices, vol. 2, MIT Press, pp. 338-372, 1964.

Allen F. and R. Karjalainen, "Using Genetic Algorithms to Find Technical Trading Rules", *Journal of Financial Economics 51*, 245-271, 1999.

Baker J.E., "Adaptive Selection Methods for Genetic Algorithms", in Proceedings of the First Int. Conf. On Genetic Algorithms and their Applications, Lawrence Erlbaum Hillsdale, NJ, pp. 101-111, 1985.

Baker J.E., "Reducing Bias and Inefficiency in the Selection Algorithm", in Proceedings of the Second Int. Conf. On Genetic Algorithms, Lawrence Erlbaum, Hillsdale, NJ, pp. 14-21, 1987.

Bauer R.J., *Genetic Algorithms and Investment*, Wiley, 1994.

Benyahia I. and J.-Y. Potvin, "Decision Support for Vehicle Dispatching using Genetic Programming", *IEEE Transactions on Systems, Man and Cybernetics 28*, 306-314, 1998.

Brock W., J. Lakonishok and B. LeBaron, "Simple Technical Trading Rules and the Stochastic Properties of Stock Returns", *Journal of Finance 47*, 1731-1764, 1992.

Bessembinder H. and K. Chan, "The Profitability of Technical Trading Rules in the Asian Stock Markets", *Pacific Basin Finance Journal*, July, 257-284, 1995.

Chambers L. (Ed.), *Practical Handbook of Genetic Algorithms: Applications*, CRC Press, Boca Raton, FL, 1995.

Chen S.-H. and C.-H. Yeh, "Toward a Computable Approach to the Efficient Market Hypothesis : An Application of Genetic Programming", *Journal of Economic Dynamics & Control*, June 1997.

Davis L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New-York, NY, 1991.

Fama E.F., "Efficient Capital Markets : a Review of Theory and Empiricial Work", *Journal of Finance 25*, 383-417, 1970.

Fama E.F. and M.E. Blume, "Filter Rules and Stock Market Trading", *Journal of Business 39*, 226-241, 1966.

Goldberg D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

Holland J.H., Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, MI, 1975.

Jensen M. and G. Bennington, "Random Walks and Technical Theories: Some Additional Evidences", *Journal of Finance 25*, 469-482, 1970.

Koza J.R., Genetic Programming: On the Programming of Computers by means of Natural Selection, The MIT Press, Cambridge, MA, 1992.

Koza J.R., Genetic Programming II: Automatic Discovery of Reusable Programs, The MIT Press, Cambridge, MA, 1994.

Koza J.R., F.H. Bennett III, D. Andre and M.A. Keane, Genetic Programming III: Darwinian Invention and Problem Solving, Morgan Kaufmann, San Francisco, CA, 1999.

Kumar A. and Y.P. Gupta (Eds.), *Computers & Operations Research*, vol. 22 (special issue on genetic algorithms), pp. 1-157, 1995.

Levich R. and L. Thomas, "The Significance of Technical Trading Rule Profits in the Foreign Exchange Market : A Bootstrap Approach", Journal of International Money and Finance 12, 451-474, 1993.

Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs, Third Edition*, Springer, Berlin, 1996.

Neely C. and P. Weller, "Technical Trading Rules in the European Monetary System", *Journal of International Money and Finance 18*, 1999.

Neely C., P. Weller and R. Dittmar, "Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach", *Journal of Financial and Quantitative Analysis 32*, 405-426, 1997.

Osler C.L. and P.H.K. Chang, "Head and Shoulders : Not Just a Flaky Pattern", Staff Paper No. 4, Federal Reserve Bank of New York, 1995.

Pruitt S.W. and R.E. White, "The CRISMA Trading System : Who Says Technical Analysis Can't Beat the Market?", *Journal of Portfolio Management 14*, 55-58, 1988.

Sweeney R.J., "Beating the Foreign Exchange Market", *Journal of Finance 41*, 163-182, 1986.

Whitley D., "The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best", in Proceedings of the Third Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, pp. 116-121, 1989.